

**MULTIMEDIA**



**UNIVERSITY**

**STUDENT ID NO**

--	--	--	--	--	--	--	--

# **MULTIMEDIA UNIVERSITY**

## **FINAL EXAMINATION**

**TRIMESTER 1, 2016/2017**

### **ECP4136 – JAVA TECHNOLOGY**

**(All Sections / Groups)**

**12 OCTOBER 2016  
9.00 a.m. – 11.00 a.m.  
(2 Hours)- OPEN BOOK**

---

#### **INSTRUCTIONS TO STUDENT**

1. This examination paper consists of FIVE (5) pages including cover with THREE (3) questions only.
2. Answer ALL questions. The distribution of the marks for each question is given.
3. This is an Open-Book examination.
4. Please write all your answers in the Answer Booklet provided.

**Instructions:**

Figure 1 shows the UML diagram of **Drawable** interface, **Rectangle** class, **Oval** class and **Line** class.

You are to "implement" the UML diagram in **Question 1** and "apply" the UML diagram in **Question 2** and **Question 3**.

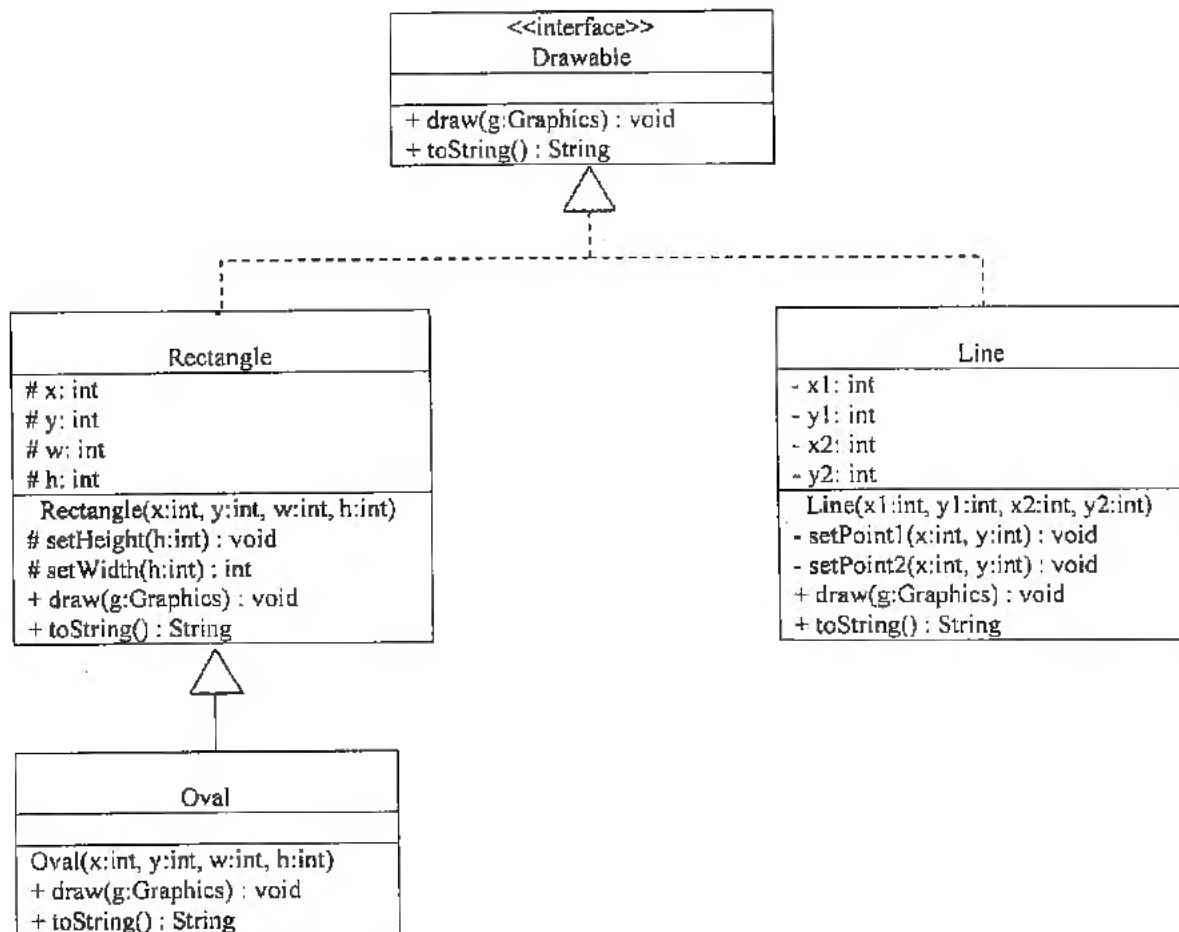


Figure 1

Continued...

**Question 1**

Implement the UML diagram shown in Figure 1. The marks distribution is as follows:

Creation of **Drawable** interface and its corresponding methods. [2 marks]

Implementation of the **Rectangle** class with the following requirements:

- Declaration of **x**, **y**, **w**, and **h** variables. [1 mark]
- Creation of a constructor that will save all parameters in arguments. [2 marks]
- Creation of a **setWidth** and **setHeight** method that will save the arguments if it is positive value. [2 marks]
- Implementing the **draw** method by drawing a rectangle with coordinate (**x**, **y**), the upper left hand corner coordinate, and values (**w**, **h**), the width and height of the shape. [1 mark]
- Implementation of **toString** method whereby a rectangle object with coordinate (2, 34) and size (56 x 789) will return "drawRect 2 34 56 789". [1 mark]

Implementation of the **Oval** class with the following requirements:

- Creation of a constructor that delegates the task to super class. [1 mark]
- Implementing the **draw** method by drawing an oval with coordinate (**x**, **y**), the upper left hand corner coordinate, and values (**w**, **h**), the width and height of the shape. [1 mark]
- Implementation of **toString** method whereby an oval object with coordinate (3, 45) and size (67 x 123) will return "drawOval 3 45 67 123". [1 mark]

Implementation of the **Line** class with the following requirements:

- Declaration of coordinate (**x1**, **y1**), and coordinate (**x2**, **y2**). [1 mark]
- Creation of a constructor that will save all parameters in arguments. Swap the two coordinates if **x2**<**x1**. When **x1**=**x2** then swap the two coordinates only if **y2**<**y1**. [3 marks]
- Creation of a **setPoint1** and **setPoint2** method that will save the parameters. [2 marks]
- Implementation the **draw** method by drawing a line between coordinate (**x1**, **y1**) and coordinate (**x2**, **y2**). [1 mark]
- Implementation of **toString** method whereby a line object with first coordinate (12, 34) and second coordinate (56, 78) will return "drawLine 12 34 56 78". [1 mark]

Continued...

## Question 2

Create an **InputOutput** class that uses a text file to read and write a series of **Drawable** object. Sample content of a file is shown in **Figure 2**. The class has following specifications:

Create a static **+write(list:ArrayList<Drawable>, filename:String):void** method that throws **IOException**. The marks distribution is as follows:

- Correct declaration of method. [1 mark]
- Creation of **PrintWriter** object. [1 mark]
- Iteration and printing the **Drawable** object into the output file. [2 marks]
- Closing the **PrintWriter** object. [1 mark]

Create a static **+read(filename:String):ArrayList<Drawable>** method that throws **IOException**. The marks distribution is as follows:

- Creation of a **Scanner** object using the filename provided and an **ArrayList** container that stores **Drawable** object. [2 marks]
- Using **Scanner** object to iterates through the files line by line, followed by word by word. [2 marks]
- Capturing the drawing action and the four parameters. [2 marks]
- Creating the appropriate object and inserting it into the **ArrayList** object. [4 marks]

```
drawline 10 10 20 20
drawRect 20 20 30 30
drawOval 30 30 40 40
drawRect 30 40 10 30
drawOval 50 30 20 20
drawline 20 10 40 20
```

Figure 2

Continued...

### Question 3

Create the program shown in **Figure 3**. The program is created using a **DrawingPanel** class. The specification of the class is as follows:

It has the following instance variables: [3 marks]

- **list**: **ArrayList**<**Drawable**> object for holding a list of **Drawable** object
- **canvasPanel**: **CanvasPanel** for drawing the **Drawable** object
- **leftPanel**: **JPanel** object for holding the **JLabel** and **JList**.
- **label**: **JLabel** for showing the text "Select shape to highlight"
- **selection**: **JList** a **JList** object for showing the list of **Drawable** object
- **index**: **int** that points to the current selected index in **JList**. Default is -1;

A constructor that receives **ArrayList**<**Drawable**> objects and perform the following action:

- Save a copy of the **ArrayList** argument and use it for the entries of **selection** object [1 mark]
- Insert **label** and **selection** object into **leftPanel**. Use the size (200 x 300) for the **leftPanel**, size (300 x 300) for the **canvasPanel** and size (500 x 300) for the current panel. Set the appropriate layout so that the **leftPanel** appear on the west and the **canvasPanel** on the center. [4 marks]
- Initialize **selection** object to be single selection mode and add a listener object such that when an item in the **selection** object is selected, the **index** is saved and **canvasPanel** is repainted [3 marks]

It has an inner class **CanvasPanel** with **paintComponent** method that will iterate through the **list** object and call the **draw** method. Finally, **index** is used to highlight the selected **drawable** object by drawing the shape using red color. [4 marks]

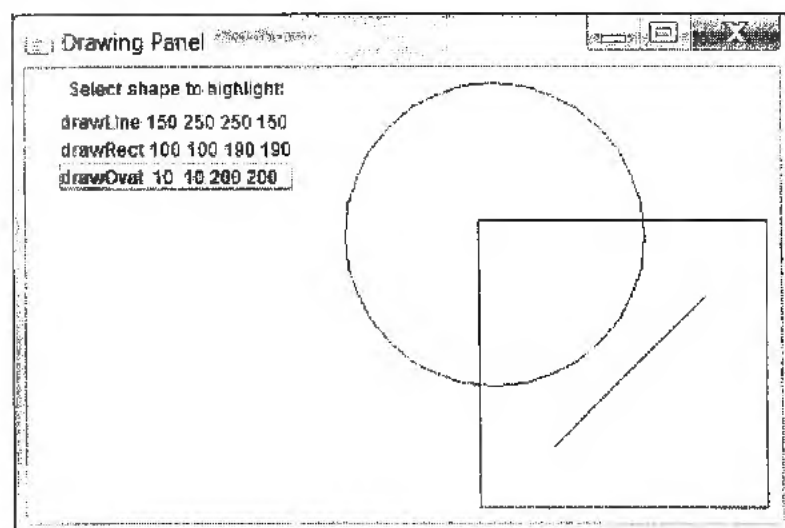


Figure 3

End of Paper